Mirage: A Microeconomic Resource Allocation System for Sensornet Testbeds

Mirage: A Microeconomic Resource Allocation System for Sensornet Testbeds

Brent N. Chun* Philip Buonadonna* Alvin AuYoung‡ Chaki Ng† David C. Parkes†
Jeffrey Shneidman† Alex C. Snoeren‡ Amin Vahdat‡

†*Harvard* **Intel Research Berkeley* ‡*UCSD*
https://mirage.berkeley.intel-research.net

Abstract

SensorNet testbeds are critical for understanding and meeting the technical challenges of wireless SensorNets. As the size and demand for these testbeds grow, resource management will become increasingly important to the effectiveness of these environments. In this paper, we argue that a microeconomic resource allocation scheme, specifically the combinatorial auction, is well suited to testbed resource management. To demonstrate this, we present the Mirage resource allocation system. In Mirage, testbed resources are allocated using a repeated combinatorial auction within a closed virtual currency environment. Users compete for testbed resources by submitting bids which specify resource combinations of interest in space/time (e.g., "any 32 MICA2 motes for 8 hours anytime in the next three days") along with a maximum value amount the user is willing to pay. A combinatorial auction is then periodically run to determine the winning bids based on supply and demand while maximizing aggregate utility delivered to users. We have implemented a fully functional and secure prototype of Mirage and have been operating it in daily use for approximately four months on Intel Research Berkeley's 148-mote SensorNet testbed.

1 Introduction

SensorNet testbeds are critical for understanding and meeting the technical challenges of wireless SensorNets. Such testbeds provide a means for developing and evaluating SensorNet technology in a controlled and instrumented environment. A canonical testbed incorporates a collection of compute and sensing nodes that are coupled together by an out-of-band communication channel and a power supply. This channel provides for remote control, reprogramming and data collection independent of a node's wireless capabilities. The power supply eliminates the need to replace batteries, reducing the physical maintenance needs of the testbed.

As with any large-scale system, resource management becomes a key aspect of testbed operation. Frequently, the infrastructure of a large testbed represents a significant capital cost to build and operate. Thus, it becomes more economical to share the testbed among users. The wholesale allocation of an entire testbed's resources to a single application or user is not desirable and frequently unnecessary. A method for partitioning the testbed based on user requirements provides a more efficient means of operation.

Current schemes for testbed resource allocation suffer from key shortcomings, falling short of meeting real user demands on a large system. First, they either lack or have inadequate mechanisms for resolving contention among competing users during times of peak demand. This often requires direct system administrator intervention to resolve conflicts. Second, they provide limited mechanisms for expressing resources, making it difficult for users to express desired resources and their associated constraints. This, in turn, affects the efficient utilization of the underlying resources by limiting the system's ability to make intelligent allocations for multiple users.

We argue that microeconomic resource allocation using a combinatorial auction is well suited to address these issues of SensorNet testbed resource management. To demonstrate this, we have implemented the Mirage testbed management system. In Mirage, resources are allocated using a combinatorial auction [3, 11]. Users submit bids specifying resource *combinations* of interest in space/time (e.g., "any 32 MICA2 motes for 8 hours anytime in the next three days") along with a maximum *value* amount the user is willing to pay. A combinatorial auction is then periodically run to determine the winning bids based on supply and demand while aiming to maximize aggregate delivered utility. Given that testbed users are typically interested in collections of nodes and are often indifferent to different allocations (as long as they meet the user's constraints) and averse to partial allocations, we believe that a combinatorial auction provides a more effective means for testbed resource allocation compared to alternatives.

In this paper, we describe the motivation, design, and implementation of Mirage. We also present details on our experience to date with a Mirage deployment on a 148-mote SensorNet testbed at Intel Research Berkeley (IRB), where Mirage currently serves as the sole means of getting physical access to testbed resources. Given the desirability of testbeds for SensorNet development and the growing user community for such testbeds, our deployment serves as a nice

"laboratory" for studying how beneficial microeconomic approaches to resource allocation are relative to all of the theoretical/algorithmic work that has gone on in this space in the past. Our primary technical contributions in this paper include:

- The design and implementation of a microeconomic resource allocation system for SensorNet testbeds that is part of a fully functional and deployed system. While previous work has largely explored the theoretical and algorithmic nature of microeconomic-based systems or presented "paper designs" of such systems, we believe that experience with working prototypes and real-world deployments are fundamental to shaping the research agenda in this space.
- A practical virtual currency policy with two novel components: (i) proportional-share *profit sharing*, to allow idle users to accumulate transient credit and (ii) a *savings tax*, which implements a "use it or lose it" policy to address highly imbalanced usage patterns. While microeconomic resource allocation has been explored in previous work, the majority of these efforts have not addressed this issue, opting instead to focus on either specific subsets of microeconomic system design or assuming the use of real money.
- Experience with an ongoing real-world deployment of our system on a 148-mote SensorNet testbed. Early experience over a four-month period indicates that demand for testbed resources is bursty, that users place significantly different value on these resources, and points to evidence of strategic user behavior. Feedback from users has also validated some of our early intuition about features (e.g., the need for sets of motes that meet specific physical constraints) of the problem that make SensorNet resource allocation non-trivial and, guided by real user needs, has pointed to future research directions.

The rest of this paper is organized as follows. In Section 2, we provide background and motivation on the testbed resource allocation problem. In Section 3, we present the design of Mirage, a microeconomic resource allocation system aimed to address this problem. In Section 4, we describe our prototype implementation. In Section 5, we describe our experience to date with real usage on an ongoing Mirage deployment on a 148-mote SensorNet testbed at Intel Research Berkeley. In Section 6, we describe related work and finally, in Section 7, we conclude the paper and describe future work.

2 Background and Motivation

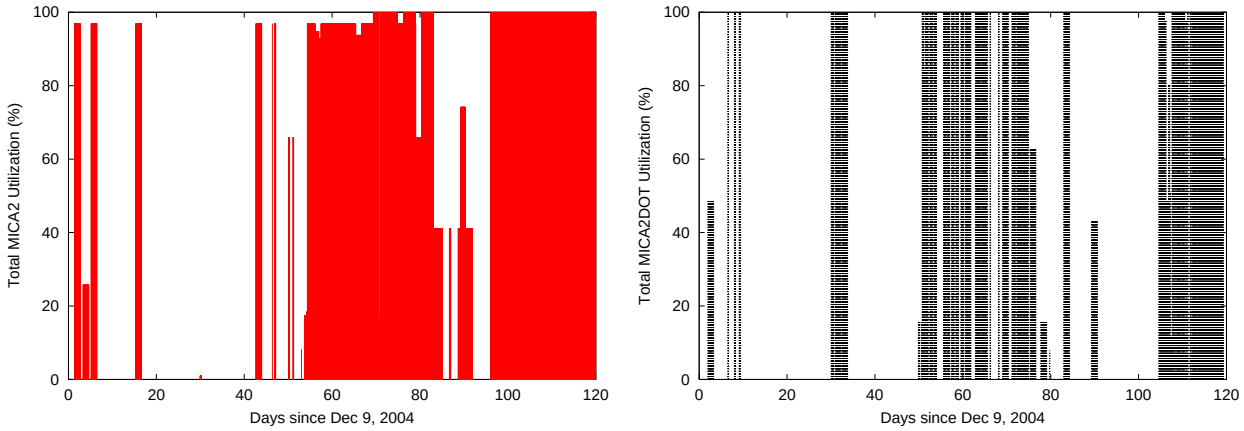
The initial motivation for this work became apparent during the construction of a 148-node SensorNet testbed at the Intel Research laboratory in Berkeley, CA. This testbed is comprised of 97 Crossbow MICA2 and 51 Crossbow MICA2DOT series sensor nodes, or motes, mounted uniformly in the ceiling of the lab. The motes incorporate an Atmel ATmega128 8-bit microcontroller, 4KB of RAM, 128KB of flash memory, and a Chipcon CC1000 FSK radio chip. The MICA2 series devices in the testbed operate in the 433 MHz ISM band and incorporate a sophisticated sensorboard that can monitor pressure, temperature, light, and humidity. The MICA2DOT devices operate in the 916MHz ISM band and do not include sensorboards.

The motes are coupled to the lab's wired production network using the Crossbow MIB600 Ethernet programming board. This device is based on a design developed at Intel Research which provides power, remote programming and out-of-band monitoring/debugging though a TCP/IP over Ethernet channel. Power is provided to each mote via the MIB600 using IEEE 802.3af power over Ethernet and can be centrally controlled from the network switch. The capabilities of the MIB600 are useful for creating large testbeds with minimal infrastructure and upkeep requirements. When the testbed was placed in service, the demand for its resources by several users was immediate, motivating the need for an effective management system.

2.1 The Case for Auction-based Resource Allocation

When contention for shared resources arises, a fundamental question is how to prioritize different users vying for access to common resources. Complicating matters further are tendencies for testbed system usage to be bursty and for users to have differential value on the resources being contended for (e.g., using resources for testing vs. using resources to perform a critical experiment leading up to a conference deadline). In Figure 1, we show testbed utilization for 97 MICA2 motes and 51 MICA2DOT motes on a 148-mote SensorNet testbed at Intel Research Berkeley over a period of four months. Examining the graphs, we see that demand for testbed resources is very bursty, with noticeable bursts in early December, early February near the SIGCOMM '05 conference deadline, and in late March / early April near both the NSDI '05 camera-ready deadline and the ACM SenSys '05 conference deadline. Further, as described in Section 5, the value (as determined by an auction) that users placed on allocations over this four month period varied over four orders of magnitude, suggesting highly varying levels of immediacy and importance in testbed usage.

The observed usage characteristics of the testbed suggest two requirements for a resource allocation system. First, there needs to be a systematic way to prioritize resource requests and second, users should be able to express the importance of their requests to use the testbed. The first requirement is motivated by the fact that testbed usage is bursty and that contention is a real phenomena, often occurring at the most important of times. The second requirement is driven by the fact that user valuations for resources varies significantly (Section 5) and



(a) MICA2 utilization

(b) MICA2DOT utilization

Figure 1: **Testbed utilization on the Intel Berkeley Lab's 148-mote SensorNet testbed from Dec 9, 2004 to Apr 8, 2005.**

thus, computing an efficient (i.e., a socially optimal allocation in terms of aggregate delivered utility) allocation requires additional information.

These requirements point towards the use of an auction-based resource allocation system. Owing to its use as a bidding process, auctions are ideally suited for eliciting the widely varying valuations that testbed users have. The ability to express a valuation explicitly combined with the fact that users bid using finite currency addresses both of our requirements. When testbed demand is high, users with the highest valuations get priority (using currency that was accumulated by making judicious use of testbed resources during previous periods of contention). When users place varying levels of importance on their requests, they simply bid higher or lower as needed. Contrast this to other scheduling schemes such as first-come first-serve (FCFS), fair-share, etc. which provide no means to explicitly express the importance of a request and thus, fundamentally operate with strictly less information, delivering less overall utility.

2.2 Combinatorics of SensorNet Testbed Allocation

Users of a SensorNet testbed are frequently interested in acquiring combinations of resources that simultaneously meet certain constraints. Consider a machine learning researcher interested in testing distributed inference algorithms in sensor networks. Such a user might be interested in evaluating algorithms at a moderate scale while performing inference over temperature and humidity readings of the environment. The user's code might also be tailored to a particular type of device (e.g., a MICA2 mote) and need to run on a different, appropriately-spaced frequency to avoid crosstalk from other experiments. Thus, the user's abstract resource specification might call for "any 64 MICA2 motes, operating on an unused frequency, that have both a temperature and a humidity sensor".

The resource combinations that users of a SensorNet testbed are interested in often have the property that there are both *substitutes* and *complementaries*. In the machine learning example above, for instance, the user is not concerned with the specific allocation of MICA2 motes as long as a total of 64 are available. Hence, MICA2 motes are substitutes for one another. Similarly, the user *does* care that 64 motes are allocated simultaneously. A partial allocation of, say, 8 motes would not meet the user's needs in this case since the user's intention was to test at a moderate scale. Thus, the 64 motes can be viewed as being complimentary to one another.

Resource allocation problems involving substitutes and complementaries are well-suited to a combinatorial auction. Users submit bids specifying desired resource combinations along with a maximum amount the user is willing to pay. The auction, in turn, computes a set of winning bids that maximizes total revenue. Since users use virtual currency to express value, the allocation that maximizes revenue collected maximizes the aggregate stated value delivered to users of the system. Given a sufficiently expressive bidding language, user preferences on both substitutes and complementaries can be captured and hence more efficient global allocations can be achieved using a combinatorial auction. In Mirage, we use a repeated combinatorial auction to periodically allocate testbed resources both in space and time.

3 Mirage

In this section, we present the design of Mirage, a microeconomic resource allocation system for SensorNet testbeds. We assume each user has a value associated with a desired resource allocation and the primary goal of the system is to maximize *aggregate value*. To maximize aggregate value, Mirage

relies on a repeated combinatorial auction [3, 11]. In such an auction, users submit bids specifying resource *combinations* of interest and the amount of virtual currency the user is willing to pay. Periodically, the auction clears, a set of winning bids is computed, and trades are settled through payments to a central bank. In this section, we describe the three main components that make up Mirage: the resource discovery service, the repeated combinatorial auction, and the central bank.

3.1 Resource Discovery

In Mirage, users specify the type of resources they are interested in using abstract resource specifications. A resource discovery service then maps these specifications to concrete resource sets that meet the desired constraints. We use this level of indirection for three reasons. First, it frees the user from having to manually identify candidate resources. Second, it allows users to automatically take advantage of new resources as they are introduced into the system. Finally, testbed users frequently wish to acquire sets of nodes but are indifferent to the specific nodes allocated as long as they meet user constraints. The resource discovery service allows users to discover all possible candidates and thus provide the system with the maximal amount of information on substitutes when clearing the auction.

Abstract resource specifications allow users to specify constraints on the types of resources they seek to acquire. For example, testbed users often need to specify constraints on per-node attributes. In other cases, constraints on inter-node attributes may be necessary. For example, a user might wish to acquire "8 motes where each pair of motes is at least 10 meters apart" to ensure that communication requires a multi-hop routing layer. Currently, our prototype supports resource discovery using per-node attributes including mote type, sensor board type, and supported frequency range. We are currently investigating how best to incorporate constraints on inter-node attributes into the system.

3.2 Repeated Combinatorial Auction

Mirage uses a first-price, repeated combinatorial auction to allocate resources to competing users over time. In this setting, an auction is run periodically. During each round, there are multiple buyers (the competing users) and a single seller who sells resources on the system's behalf. All bids submitted prior to the start of a round are considered. The auction will then calculate winning bids, their payments, and associated resource allocations.

Users submit bids to the auction using a two-phase process (Figure 2). First, they use the resource discovery service to find candidate nodes that meet their constraints. Second, using concrete nodes identified from the first step, they place bids in the auction using the Mirage bidding language.

Since time is a critical aspect of resource allocation (e.g., resources near a conference deadline), resource combinations

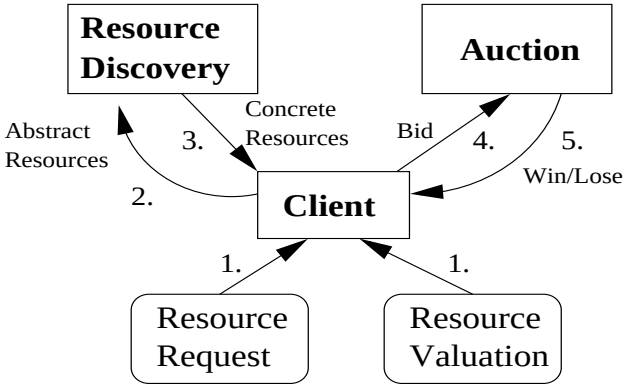


Figure 2: **Bidding and Acquiring Resources.**

specify resources in both space and time. Formally, a bid b_i in Mirage is specified as follows:

$$b_i = (v_i, s_i, t_i, d_i, f_{min}, f_{max}, n_i, ok_i) \quad (1)$$

Bid b_i indicates the user wants *any combination* of n_i motes from the set ok_i (obtained through resource discovery) for a duration of d_i hours, a start time anywhere between s_i and t_i, and a frequency in the range $[f_{min}, f_{max}]$. The user also is willing to pay up to v_i units of virtual currency for these resources. Continuing with the distributed inference example, a user thus might say: "any 64 MICA2 motes, which have both a temperature and a humidity sensor, operating on an unused frequency in the range [423 MHz, 443 MHz], for 4 consecutive hours anytime in the next 24 hours". Suppose the user used the resource discovery service and found 128 motes meeting the desired resource specification and valued the allocation at 99 units of virtual currency. The corresponding bid in this case would thus be:

$$b_i = (99, 0, 20, 4, 423, 443, 64, \text{list of 128 motes}) \quad (2)$$

The goal of the system in each round is to clear the auction such that the winning bids maximize the amount of revenue collected by the system. Since users use currency to express value, the allocation that maximizes revenue maximizes the aggregate delivered value and thus can be viewed as being socially optimal. This problem of computing a revenue-maximizing allocation in a combinatorial auction is known as the winner determination problem and is known to be NP-hard by reduction from weighted set packing [14]. Thus, in our prototype, we currently use a greedy heuristic to compute the set of winning bids. We are also currently investigating a mixed-integer program formulation using CPLEX, a commercial, linear-programming based, branch-and-cut solver for mixed integer programs.

3.3 Central Bank

Mirage supports virtual currency, because charging real currency in our environment is impractical. To support virtual currency, Mirage relies on a central bank to enforce currency policy. A good virtual currency policy is instrumental in controlling the aggregate amount and flow of virtual currency in the system, and to encourage desirable behaviors, while discouraging undesirable ones. The lack of proper policies can render the system useless. For example, if users can obtain large amounts of virtual currency easily, then they will bid arbitrarily high values at all times. Such a system reduces to resource allocation based on social conventions since there is no disincentive for a user to not always bid the maximum possible value.

Mirage is a closed economic system and users have no way to earn currency. Thus, the system must decide how to distribute virtual currency. Because users enter the system with no virtual currency, we must bootstrap users into the system by providing them with some amount of initial currency. In addition, as users spend currency over time, we also need a way to infuse their accounts with new currency. Clearly, there are many currency polices to meet these requirements and different policies will result in different economic systems and resource allocations.

Our virtual currency policy is based on two principles: (i) prioritizing users based on type of usage and (ii) penalizing/rewarding users based on usage during times of peak demand. Prioritizing based on type of usage (e.g., research vs. coursework) is a reasonable strategy lacking other metrics to differentiate users. In addition, it seems natural to reward the user who refrains from using the system during times of peak demand (or, more generally, does not waste resources) and penalize the user who uses resources aggressively when resources are most scarce.

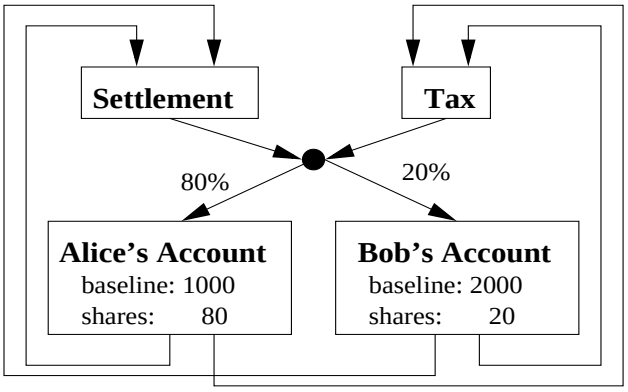


Figure 3: **Virtual currency policy.**

In Mirage, each user is associated with a project that has an account at the central bank. Each project's bank account is assigned a baseline amount of currency based on priority, and a number of currency shares which influence the rate that currency flows into the account. Given initial currency, users can bid for resources in the auction. Each time the auction clears, bids are settled and revenue is collected from the accounts for winning bids. This currency is then distributed back to all accounts through profit sharing in a proportional-share fashion based on the number of shares in each account (Figure 3). It is this profit sharing policy that allows users who do not waste resources to save additional currency which can be used for a subsequent burst of activity.

In addition to profit sharing, the system also imposes a fixed rate savings tax on all accounts that have excess currency above their baseline values, again with proportional-share distribution. The motivation for the savings tax is based on expected resource consumption. For example, in other distributed systems testbeds such as PlanetLab [12], it has been observed [1] that resource consumption is often highly imbalanced with a small fraction of users consuming the majority of the resources and many users often going idle for long periods of time. Given similar resource patterns here and in the absence of additional policy, the implication would be that heavy users would eventually be working out of accounts with very little currency even if there is low demand for testbed resources. To mitigate this effect, we thus impose a fixed rate savings tax (the "use it or lose it" policy). The basic idea here is that users should be rewarded for not wasting resources but that such a reward should not last forever. In the absence of any activity in the system, the savings tax works such that all accounts eventually converge back to their baseline values.

4 Implementation

We have implemented a prototype of Mirage and have been operating it for approximately four months on Intel Research Berkeley's 148-mote SensorNet testbed. The implementation is comprised of three types of components: clients, a server, and a front-end machine that provides controlled physical access to the testbed (Figure 4). Clients provide users with secure, authenticated command-line (the `mirage` program) and web-based access to a server (`miraged`) which implements a logical combinatorial auction, bank, and resource discovery service. The server accepts secure, authenticated XML-RPC requests using the SSL protocol with persistent state stored in a PostgreSQL database. Lastly, the front-end physically enforces resource allocations from the auction using Linux's per-uid `iptables` packet filtering capabilities. By default, all users are denied access to all motes. Based on the outcome of the auction, rules are added as needed to open access to users of winning bids for specific periods of time.

Several variables parameterize the auction: the number of resource slots, resource slot size, and acceptable bid durations. Our initial parameterization is based on expectations of user desires but will likely evolve. Access to the testbed is based on 1-hour slots. To accommodate users who require a range of

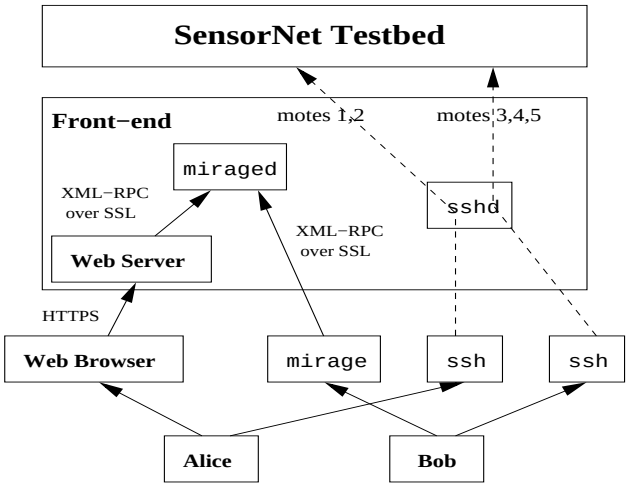


Figure 4: **Mirage implementation.**

different time slots, users may bid for 1, 2, 4, 8, 16, or 32 hour duration blocks. To allow users to plan ahead, the auction sells resources up to three days in advance. Given our 1-hour slot size, this works out to a total of 72 slots. Thus, we can view the resources being allocated as a matrix of 148 motes by 72 slots. When the system boots, all slots are available. Over time, slots become occupied as bids are allocated and new slots become available as the window of slots opens up over time.

To use the system, users register for an account at a secure web site by providing identifying information, contact information, a project name, and by uploading an SSH public key. Each user is associated with a project and each project has an owner. An administrative user is responsible for enabling accounts for project owners and assigning each project a baseline virtual currency value and a number of virtual currency shares. Project owners can subsequently enable their own users, thereby eliminating the administrative user as a centralized bottleneck. In our deployment, most projects have baseline and shares values set to 1000. Two projects (`bbq` and `dcs`), involving local users at the lab, have larger allocations with baseline and shares values set to 2000.

Once enabled, users can securely bid in the auction using either the command-line tool `mirage` or through the web interface, where PHP scripts on the backend act as XML-RPC/SSL clients to the `miraged` server. The command-line tool provides access to the entire RPC interface exposed by `miraged`. Use of this program is useful for scripting and automation. To accommodate users who prefer a graphical interface, the web-based interface provides a simple, integrated interface where users specify what resources they want and how much they are willing to pay using an HTML form. The web server, in turn, maps the user's abstract resources to concrete resources using the resource discovery service and places a bid in the auction on the user's behalf.

For winning bids, Mirage provides access to the associated set of motes for the associated amount of time to project members of the winning bid. Mirage provides physical access to each user by: (i) creating a temporary Unix login on the front-end machine using a global username (the base32-encoded MD5 hash of the user's SSH public key), (ii) enabling access to the front-end via SSH authentication using an SSH `authorized_keys` file, and (iii) setting up firewall rules on the front-end such that only the user can access the particular motes assigned to the winning bid.

5 Usage and Experience

We have been operating Mirage on Intel Research Berkeley's 148-mote SensorNet testbed for approximately four months now. As of April 4, 2005, users from 18 different research projects have registered to use the system and 322 bids have been submitted resulting in a total of 312148 allocated node hours. While our experience to date is still preliminary given the length of time the system has been in operation, measurements of system usage thus far point to three findings: (i) demand for testbed resources is bursty (Figure 1), (ii) users place significantly different value on resources, varying over four orders of magnitude, and (iii) there is evidence suggesting that some users are behaving strategically. These characteristics point toward the applicability of an auction for resource allocation in SensorNet testbeds.

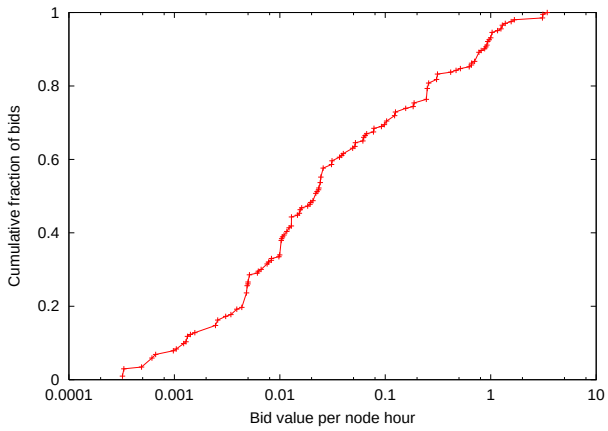


Figure 5: **Cumulative distribution of bid values per node hour.**

In Figure 5, we show the cumulative distribution function (CDF) of bid values per node hour for all bids submitted to the system. A bid's value per node hour is computed as follows: if a bid has value v and requires n nodes for h hours, then that bid's value per node hour is v/nh. From the graph, we observe that users place significantly different value on testbed resources, varying over four orders of magnitude from a value of 0.000332 per node hour to a value of 3.43 per node hour, a factor of 10060 difference. We also see that these widely

varying valuations are distributed relatively evenly across each order of magnitude, suggesting that this range is not due to a few anomalous bids but rather to a wide range of underlying user valuations for testbed resources. These two observations support the use of auctions, which are designed precisely to elicit such widely varying valuations to compute an efficient allocation.

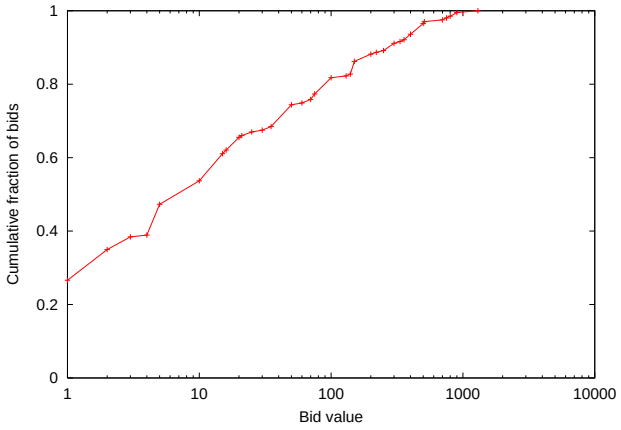


Figure 6: **Cumulative distribution of bid values.**

Node hours	Project
113827 (36.47%)	distreg
62015 (19.87%)	tinierdb
34913 (11.18%)	snetrouting
26161 (8.38%)	bbq
24831 (7.95%)	dcs
14758 (4.73%)	rbrouting
10940 (3.50%)	xmesh
10603 (3.40%)	rads
8124 (2.60%)	tinyos
5464 (1.75%)	ucd
512 (0.16%)	nucleus
0 (0.00%)	tinydb
0 (0.00%)	racelab
0 (0.00%)	princeton
0 (0.00%)	harvard
0 (0.00%)	fps
0 (0.00%)	epfl
0 (0.00%)	bbq_routing

Table 1: **Node hours consumed distributed by project.**

Figure 6 shows the same information except without the node hours scaling factor. It shows the distribution of bid values as entered by users when bidding through Mirage's web interface. Again, we see that users make significant use of the ability to express a wide range of valuations for testbed resources. This graph indicates that the value of desired allocations, regardless of how many nodes or hours were required, spanned three orders of magnitude.

In Table 1, we show aggregate node hours consumed by each of the active projects using the system. The data indicates that system usage varies widely across projects and is highly imbalanced with the top 22% of projects (the top four projects) consuming over 75% of aggregate node hours delivered. The `distreg` project alone accounts for 36.47% of all node hours used in the system. In contrast, the `nucleus` and `ucd` projects combined have consumed just 1.91% of total allocated node hours and seven projects have remained idle the entire time. Given the bursty demand for system resources shown early in Figure 1, this data motivates the need for tracking consumption and for weighting such consumption by the pain it imposes on other users. For example, when contention is high, it would seem natural that `distreg`, given its previous consumption, would have a lower probability of acquiring testbed resources than, say, `ucd`. On the other hand, if most of `distreg`'s consumption occurred when the system was idle, then its associated consumption penalty should be reduced accordingly given that the resources would have gone unused otherwise. Again, these properties are ones that are well suited for an auction-based resource allocation system.

Figures 7 and 8 show CDFs for the number of nodes users requested and the lengths of time (in hours) they requested those nodes for. From the nodes CDF, we see that approximately 16% of the bids requested 32 nodes or less, resources that were likely related to development and debugging. On the high end, we see two noticeable spikes, one at 51 nodes and another at 97 nodes, corresponding to the 51 MICA2DOT nodes and 97 MICA2 nodes in the testbed. The number of bids for either all 97 MICA2 motes or all 51 MICA2DOT motes increased considerably near the SenSys '05 deadline in late March / early April. Turning to the time durations CDF, we find that approximately 40% of the bids requested allocations of 8 hours or more. At the same time, we also see that more than 40% of the bids required 4 hours or less. All of the above suggests that flexible sharing in both space and time of a SensorNet testbed is necessary.

Early system measurements also suggest that some users are behaving strategically rather than simply bidding their true value for desired resources. This suggests that moving to an auction that is *strategyproof* [5, 8, 10], where a rational user's optimal strategy is to always bid her true value, is likely to result in further efficiencies. As one example, we found one user who previously would often bid 1 for any amount of node hours requested then modify his bidding strategy accordingly when competition (i.e., other active/recent bids) was observed. Since users are allowed to modify their bids and the auction process was open (all bids were visible), such a strategy should in theory have no effect on who wins the auction since users with higher valuations (and who also may lowball their bids) should eventually outbid those with lower valuations after a sufficient amount of iteration. The problem is that users don't behave this way. Usability overhead matters—they bid once and perhaps modify their bid a second time, the end result be-

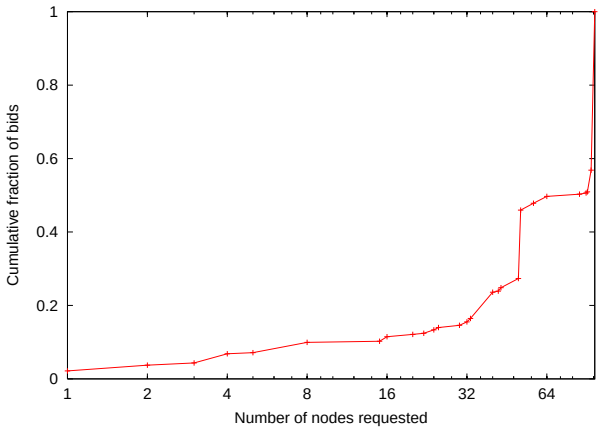


Figure 7: **Cumulative distribution of number of nodes requested.**

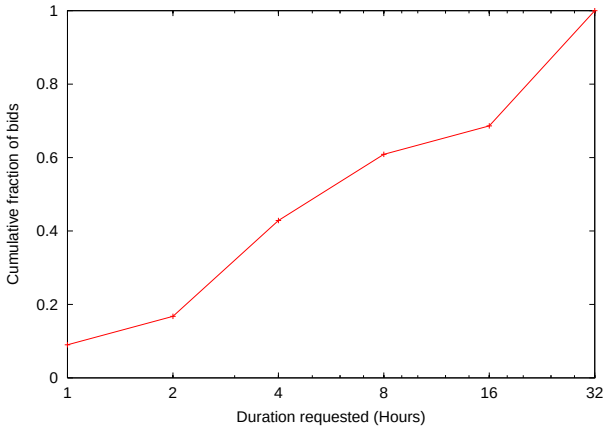


Figure 8: **Cumulative distribution of durations (in hours) requested.**

ing that inefficiencies can arise.

More problematic than this was the strategic behavior recently observed near the SenSys '05 conference deadline (April 8, 2005). The first problem occurred when the same user recently discovered a loophole in the auction's rolling window of resources, a problem that fundamentally does not exist in conventional one-shot auctions and is due entirely to the temporal nature of computational resources. Under heavy contention, it allowed a user to bid in the auction in such a way that low-value bids for shorter durations could win against high-value bids for longer durations. Upon recognizing this exploit, we deployed a new sealed-bid (i.e., current bids are not visible) auction mechanism on March 30, 2005 and publicly announced the change on the Mirage users mailing list. Unfortunately, shortly after announcing this fix, the same user discovered a different loophole, this time exploiting the greedy nature of the auction algorithm in combination with the observed workload (previous bids that have won are posted for price feedback purposes). This loophole involved the user employing a strategy of splitting his bid for 97 MICA2 motes across several bids, only one of which had a high value per node hour. Since the high value bid is likely to win due to the greedy nature of the auction clearing algorithm and since all other users at the time were all requesting 97 motes, no other bids could backfill into the remaining slots; the user's remaining bids would then fit those slots at a low price.

In summary, these problems point clearly towards the need for an auction that is either strategyproof or, minimally, hard-to-manipulate (e.g., gaming the system is NP-hard). Because testbed allocation in Mirage involves resources in space/time and combinatorial bids over a rolling window of resources, however, this mechanism design problem is non-trivial. It is clear, however, that patching an existing non-strategyproof mechanism is problematic in the presence of users desperately trying to acquire scarce resources. A key lesson learned from recent usage is that users will try their hardest to game the system precisely during the times when it is most important (i.e., under heavy demand). Consequently, we believe that moving to either a strategyproof or hard-to-manipulate auction is a fundamental requirement to maximize the efficacy of market-based systems such as Mirage. While the system today appears well-suited to the resource demands of SensorNet testbed users (i.e., widely varying valuations, bursty demand, the need for flexible allocations in space/time, etc.), marrying the system with an improved auction that has theoretically provable properties should result in even better performance.

As a final note, we point out that informal feedback from users has indicated that a resource allocation system based on a combinatorial auction need not be hard to use given the simple web interface that Mirage provides. The main problems users have experienced (besides the recent strategic exploits mentioned above) have mainly concerned mechanics of logging into the front-end and a hardware failure that caused an outage for several days in mid-January. Additional user feedback has also involved requests for new features in the system, some of which present additional research challenges. Notable requests include: (i) "buy-it-now" prices that allow resources to be acquired without having to wait for the auction to clear, (ii) bidding for resources that meet certain physical constraints, and (iii) bidding for an identical set of resources for repeatability. The last of these requests was implemented in late February 2005, while the first two will require augmenting the interface to Mirage and developing new algorithms.

6 Related Work

Originally, the lab's SensorNet testbed used the Motelab scheduling and management system [18] for testbed resource allocation. Motelab provides a simple scheduling mechanism via a web-based interface and includes automated reprogramming and data collection capabilities. Resource requests are expressed by having users select time slot(s) from a master

schedule in a first-come first-serve (FCFS) fashion and a simple time-based quota scheme is used to regulate equitable use of the testbed schedule amongst active users. As mentioned in Section 2.1, FCFS scheduling suffers from fundamental limitations that make it difficult to resolve resource contention in an efficient manner. In addition, since users request physical resources (as opposed to abstract resources specifying desired constraints) in Motelab, this further reduces opportunities for resource allocation optimizations when users are indifferent to multiple potential allocations. In our view, Motelab's primary strength is its automated reprogramming and data collection capabilities and such mechanisms are entirely complementary to Mirage.

There are a number of existing approaches to resource allocation in computing systems. Proportional-share scheduling is an approach commonly used to provide equal access to time-shared resources. This approach provides users easy access to all resources since it determines allocations using a well-defined metric. However, it does not seem appropriate for a shared testbed, and in particular, for a system in which over-demand for resources can be a common occurrence. For example, as demand for resources during these periods increases, the time-share per resource received by each user decreases, thereby lowering the utility delivered by the system to the user. A resource allocation system should be able to manage allocations during times of light and heavy resource contention.

Batch scheduling is a commonly used approach used to schedule jobs in supercomputers and grid computing environments. These systems aim to schedule jobs in a way that optimizes system-level metrics such as total throughput or average job-turnaround time. However, these systems do not take into account user preferences in scheduling jobs. For example, such batch scheduling systems cannot distinguish between jobs that users require a timely completion and jobs that do not. In order to maximize the utility a system delivers to end-users, the system needs to elicit users' preferences for resources over time and space and seek to optimize user-level metrics. [7].

A number of previous efforts have explored market-based approaches for resource allocation in computer systems. The earliest work that we are aware of is a futures market for CPU time on Harvard's PDP-1 [16]. Subsequent work has been largely dominated by auction-based schemes and been applied to resource allocation in a broad range of distributed systems including clusters [2, 17], computational Grids [6, 19], parallel computers [15], and Internet computing systems [9, 13]. As with these systems, Mirage also relies on an auction to allocate resources. A key distinguishing feature of Mirage, however, is its use of a *combinatorial* auction where users bid on bundles of resources as opposed to individual nodes. This ability to bid on resource combinations in space and time allows users to more accurately express their preferences on the distributed resources they seek to acquire. Such an ability we believe is highly desirable in a setting like SensorNet testbed allocation where users are typically interested in getting access to collections of nodes simultaneously.

7 Conclusion and Future Work

Looking forward, we believe that microeconomic resource allocation using combinatorial auctions is well suited for large, shared SensorNet testbeds. We have explored this concept through Mirage, a microeconomic resource allocation system for SensorNet testbeds. The primary goal of Mirage is to allocate testbed resources to competing users in a maximally efficient manner in terms of aggregate utility delivered to users. To accomplish this, Mirage uses a repeated combinatorial auction. Users submit bids which express desired resource combinations in space/time along with a maximum amount the user is willing to pay in units of virtual currency. A combinatorial auction then determines an efficient set of winning bids and subsequently provides users with controlled physical access to the relevant nodes. To address practical issues pertaining to expected testbed usage and the fact that Mirage is a closed economic system, Mirage employs a virtual currency policy based on baseline virtual currency amounts, profit sharing, and a savings tax. The end result of this policy is that users can be prioritized based on type of usage and users are penalized/rewarded based on usage or lack of usage during times of peak demand. Our Mirage prototype has been deployed on a 148-mote SensorNet testbed and currently serves as the sole means of getting access to the testbed in nearly four months of ongoing operation. Early experience with real usage indicates that demand for testbed resources is bursty, that users place significantly different value on these resources (varying over four orders of magnitude), and points to evidence of strategic user behavior.

We acknowledge that there are still many open questions surrounding auctions for resource allocation. While an auction may provide the highest aggregate utility, it leaves to question whether it produces the most socially optimal allocation. For SensorNet testbeds, users may also wish to adhere to etiquette that is not necessarily captured in to the auction process. For example, this might involve giving preferential access to groups of users based on seniority irrespective of the the perceived monetary value. Another question concerns whether users will actually express the true value of their bid. As discussed, we have witnessed some users employing strategies to bid minimal amounts for their allocations and exploiting weaknesses in the auction algorithm. Further investigation and development of either strategyproof or hard-to-manipulate auction mechanisms is needed to minimize opportunities for pathologic gaming of the system. Related to bidding, another open issue is whether Mirage's current bidding language is expressive enough to capture the full range of resource requests that users would like to make. Assuming extensions are needed, as initial feedback has suggested, another question is whether efficient algorithms can be designed to produce high quality allocations while simultaneously clear-

ing the auction in a timely manner. Finally, while Mirage's current virtual currency policy provides mechanisms that we believe will be effective in practice, further investigation with real workloads over a longer period of time is still needed for proper tuning and evaluation.

To address such limitations, future work on this project revolves around four primary areas. First, we intend to continue to operate Mirage and gather further experience with real users and usage over an extended time period. Based on measurement and user feedback, we intend to obtain further insights on the issues associated with making a real economic-based resource allocation system work in practice and to quantify how well the system performs over time with a real workload. Lessons learned might then be used to focus both theoretical and systems work on the most germane areas (e.g., combinatorial resource allocation with inter-node constraints). Second, we plan to develop strategyproof and/or provably hard-to-manipulate auction mechanisms to minimize the impact of strategic user behavior. Such mechanisms are directly aimed at addressing the strategic behavior we have recently observed near the SenSys '05 conference deadline. Third, we are working with other research groups that operate SensorNet testbeds elsewhere in the world on gathering user requirements and, in one case, deploying Mirage on their local testbed. Fourth, we are also investigating how to extend this work to build an open economy of federated SensorNet testbeds where each testbed has its own combinatorial auction for local resources, its own virtual currency, and testbeds trade foreign currencies using either transitive bartering [4] and/or a virtual currency exchange to access remote resources. The primary benefits of having such a federation include: access to resources not available locally (e.g., different types of motes and/or sensors), access to idle remote resources when local resources are under contention, and enabling open trading of resource rights among participants.

References

[1] B. Chun and A. Vahdat. Workload and Failure Characterization on a Large-Scale Federated Testbed. Technical Report IRB-TR-03-040, Intel Research Berkeley, November 2003.

[2] B. N. Chun and D. E. Culler. User-centric Performance Analysis of Market-based Cluster Batch Schedulers. In *Proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid*, May 2002.

[3] S. de Vries and R. V. Vohra. Combinatorial Auctions: A Survey, 2000.

[4] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. SHARP: An Architecture for Secure Resource Peering. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, October 2003.

[5] A. Kothari, D. C. Parkes, and S. Suri. Approximately-strategyproof and tractable multi-unit auctions. *Decision Support Systems*, 2004. Special issue dedicated to the 4th ACM Conference on Electronic Commerce.

[6] K. Lai, B. A. Huberman, and L. Fine. Tycoon: A Distributed Market-based Resource Allocation System. Technical Report cs.DC/0404013, April 2004. Available at `http://arxiv.org/abs/cs.DC/0404013`.

[7] C. Lee, A. Snavely, B. Leary, L. Carrington, H. Casanova, R. Bohn, R. Carson, J. Hardy, and Y. Schwartzman. Towards High-Order Performance Objectives for HPC System Scheduling. Technical report, University of California San Diego, March 2004.

[8] D. Lehmann, L. I. O'Callaghan, and Y. Shoham. Truth Revelation in Approximately Efficient Combinatorial Auctions. *Journal of the ACM*, 49(5):577–602, 2002.

[9] L. Levy, L. Blumrosen, and N. Nisan. On Line Markets for Distributed Object Services: the MAJIC System. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems*, March 2001.

[10] C. Ng, D. C. Parkes, and M. Seltzer. Virtual Worlds: Fast and Strategyproof Auctions for Dynamic Resource Allocation. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, pages 238–239, 2003. Short paper.

[11] N. Nisan. Bidding and Allocation in Combinatorial Auctions. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, October 2000.

[12] L. Peterson, D. Culler, T. Anderson, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proceedings of HotNets-I*, October 2002.

[13] O. Regev and N. Nisan. The POPCORN Market – an Online Market for Computatioal Resources. In *Proceedings of the 1st International Conference on Information and Computation Economies*, October 1998.

[14] M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally Manageable Combinational Auctions. *Management Science*, 44(8), August 1998.

[15] I. Stoica, H. Abdel-Wahab, and A. Pothen. A Microeconomic Scheduler for Parallel Computers. In *Proceedings of the 1st Workshop on Job Scheduling Strategies for Parallel Processing*, April 1995.

[16] I. E. Sutherland. A Futures Market in Computer Time. *Communications of the ACM*, 11(6):449–451, 1968.

[17] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and S. Stornetta. Spawn: A Distributed Computational Economy. *IEEE Transactions on Software Engineering*, 18(2):103–177, February 1992.

[18] M. Welsh and G. Werner-Allen. MoteLab: Harvard Sensor Network Testbed. `http://motelab.eecs.harvard.edu`.

[19] R. Wolski, J. S. Plank, T. Bryan, and J. Brevik. G-Commerce: Market Formulations Controlling Resource Allocation on the Computational Grid. In *Proceedings of the 15th International Parallel and Distributed Processing Symposium*, March 2001.